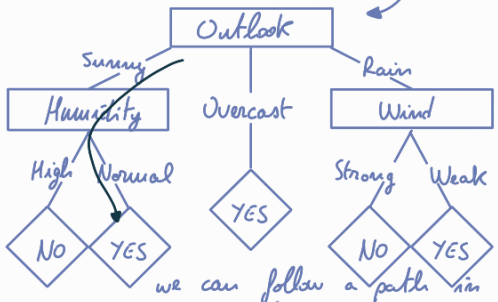
Decision trees are used to learn discrete-valued functions. These learning methods are among the most popular of inductive inference algorithms.

Deductive learning algorithms: conclusion ⟶ observation
Inductive learning algorithms: observation ⟶ conclusion ∋ decision trees

each node in the tree specifies a test of some attribute of the instance and each branch descending from that node represents one possible value for this attribute. It provides a good explaination of what is happening

Suppose we have a discrete input space described by m attributes $X = A_1 \times A_2 \times \dots \times A_m$ and a classification problem $f: X \longrightarrow C$. In this scenario, we could learn the target function by building a tree.
Decision trees classify instances by sorting them by this tree from the root to the leaf node.
Hypothesis space = set of all the possible decision trees ⟶ we will choose a single tree between them.

- The ID3 family of algorithms searches a complete hypothesis space in a incomplete way: the space itself is capable of expressing any finite discrete-valued function but the ID3 algorithms stop when, scanning from simple to complex hypotheses, the termination condition is met (e.g. when an hypothesis consistent with the data is found).
- The version space candidate-elimination algorithm searches an incomplete hypothesis space (can express only a subset of the potentially teachable concepts) in a complete way (finding every hypothesis consistent with the training data).



we can follow a path in order to classify a new instance
f(O) = Sunny, H = normal, W = *, T = ?
= YES

inductive bias → ID3 ⟶ preference for certain hypotheses over others (e.g. shorter) ⟶ search bias
→ candidate-elimination → restriction on the set of hypotheses considered ⟶ language bias

Occam's razor: prefer the simplest hypothesis that fits the data

A preference bias is usually more desirable than a restriction bias because it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function.

We can extract a rule from each decision tree: if we want a rule for YES we consider the conjunction of all the paths that lead to YES (disjunction of conjunctions of constraints on the attribute values of instances).

YES: (Outlook = Sunny ∧ Humidity = Normal) ∨ (Outlook = Overcast) ∨
(Outlook = Rain ∧ Wind = Weak)

paths from root to leaf ⟶ conjunction of attribute tests
tree ⟶ disjunction of conjunctions

Decision tree learning is generally best suited to problems with the following characteristics:

- instances are represented by a fixed (finite) set of attribute-value pairs;
- the target function has discrete output values → concept learning
  → classification
- disjunctive descriptions may be required: in this setting, all possible hypotheses are taken into account;
- the training data may contain errors and/or missing attribute values;

thus we prefer ID3 (family) over candidate-elimination

The basic ID3 algorithm build the tree top down; each time it selects the best attribute. Which is the best attribute? We have to choose an attribute that is able to better split the examples according to its value, in such a way that all the subsets more or less agree on the classification. The key idea is that we would like to select the attribute that is most useful for classifying samples.

Information gain measures how well a given attribute separates the training examples according to their target classification. ID3 uses information gain to select among the candidate attributes at each step while growing the tree. ID3 selects the attribute that induces highest information gain. Information gain is a reduction in entropy.

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

p_i = proportion of S belonging to class i

Entropy = 1 is the worst case possible ☹

Entropy = 0 is very good ☺

Entropy = 0 ⟶ all members of S belong to the same class ☺

Entropy = 1 ⟶ the collection contains an equal number of samples for each class (we can't choose) ☹

Information gain is the expected reduction in entropy of S obtained by splitting the dataset according to the value of attribute A.

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

weighted average of the entropy over S

Information gain example:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis	
1	Sunny	Hot	High	Weak	No	- W
2	Sunny	Hot	High	Strong	No	- S
3	Overcast	Hot	High	Weak	Yes	+ W
4	Rain	Mild	High	Weak	Yes	+ W
5	Rain	Cool	Normal	Weak	Yes	+ W
6	Rain	Cool	Normal	Strong	No	- S
7	Overcast	Cool	Normal	Strong	Yes	+ S
8	Sunny	Mild	High	Weak	No	- W
9	Sunny	Cool	Normal	Weak	Yes	+ W
10	Rain	Mild	Normal	Weak	Yes	+ W
11	Sunny	Mild	Normal	Strong	Yes	+ S
12	Overcast	Mild	High	Strong	Yes	+ S
13	Overcast	Hot	Normal	Weak	Yes	+ W
14	Rain	Mild	High	Strong	No	- S

Entropy:

$Values(Wind) = \{Weak, Strong\}$

$S = [9+, 5-]$ $\quad |S| = 14$

$S_W = [6+, 2-]$ $\quad |S_W| = 8$

$S_S = [3+, 3-]$ $\quad |S_S| = 6$

$$Entropy(S) = -\frac{9}{14}\log\frac{9}{14} - \frac{5}{14}\log\frac{5}{14} = 0.94$$

$$Entropy(S_W) = -\frac{6}{8}\log\frac{6}{8} - \frac{2}{8}\log\frac{2}{8} = 0.81$$

$$Entropy(S_S) = -\frac{3}{6}\log\frac{3}{6} - \frac{3}{6}\log\frac{3}{6} = 1.00$$

$$Gain(S, Wind) = Entropy(S) - \frac{8}{14} Entropy(S_W) - \frac{6}{14} Entropy(S_S)$$

$$= 0.94 - \frac{8}{14}\,0.81 - \frac{6}{14}\,1.00 = 0.048$$

heuristic = information gain

(search) bias = preference on simpler hypotheses

The Base ID3 algorithm can produce trees that overfit the training examples → overfitting means that it exists some other hypothesis that performs worse on the training examples but better on the entire distribution of instances

formal definition of overfitting

> given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$ such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances

approaches to avoid overfitting
- stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- grow the full tree, then post-prune

we can use a separate set of examples (distinct from the training examples) to evaluate the utility of post-pruning

dataset
- training: used to form the learned hypothesis
- validation: used to evaluate the accuracy of the current hypothesis, the refinement needed
- test: used to evaluate the overall hypothesis (final) of the model

Reduced-error pruning: remove the subtree rooted at a node, making it a leaf and assigning it the most common classification of the training examples affiliated with that node. Nodes are removed only if the resulting pruned tree performs no worse than the original one over the validation set

→ it produces the smallest version of most accurate subtree

warning: when data is limited, further reducing the set of training examples can be harmful and yield bad results

Rule post-pruning: we work on rules rather than nodes; prune (generalize) each rule

Decision trees can be easily extended to also use continuous valued attributes

temperature = 82.5 → temperature > 72.3 = t
temperature = 75.0 → temperature > 72.3 = t
temperature = 80.2 → temperature > 72.3 = t
... ...

temperature > 72.3 = {t, f}

GainRatio other than information gain is another possible metrics used to choose the next attribute that is particularly efficient when we have attributes with too many values (it penalizes them)

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInformation(S,A)}$$

$$SplitInformation(S,A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

→ this term is sensitive to how broadly and uniformly the attribute splits the data